GIT 소개 프레젠테이션

GIT 에 대해 알아보자

YANG MIN JUN

Contents

Git에 대한 간단한 소개와 기본 개념을 살펴보겠습니다.
Git은 협업과 버전 관리를 위한 강력한 도구로, 소프트웨어 개발에서 더 나은 효율성과 협업을 가능하게 합니다. 함께 알아보도록 하겠습니다.

Chapter 1.

서비스 소개

GIT에 대해 알아보자

GIT의 장점?

- 병렬 개발

소스 코드를 주고 받을 필요 없이,
같은 파일을 여러 명이 동시에 작업하는 병렬 개발이 가능

- 분산 버전 관리

분산 버전 관리이기 때문에 인터넷이 연결되지 않은 환경에서도
개발을 진행할 수 있다

- 체계적인 개발

GIT 통해 버전관리를 하면 체계적인 개발이 가능해지고
프로그램이나 패치를 배포하는 과정도 간단해진다.

등등 이밖에도 수많은 장점이 존재한다.

단 잘 쓸수 잇다면..

잘쓸수 있게 알아보자!

Chapter 2.

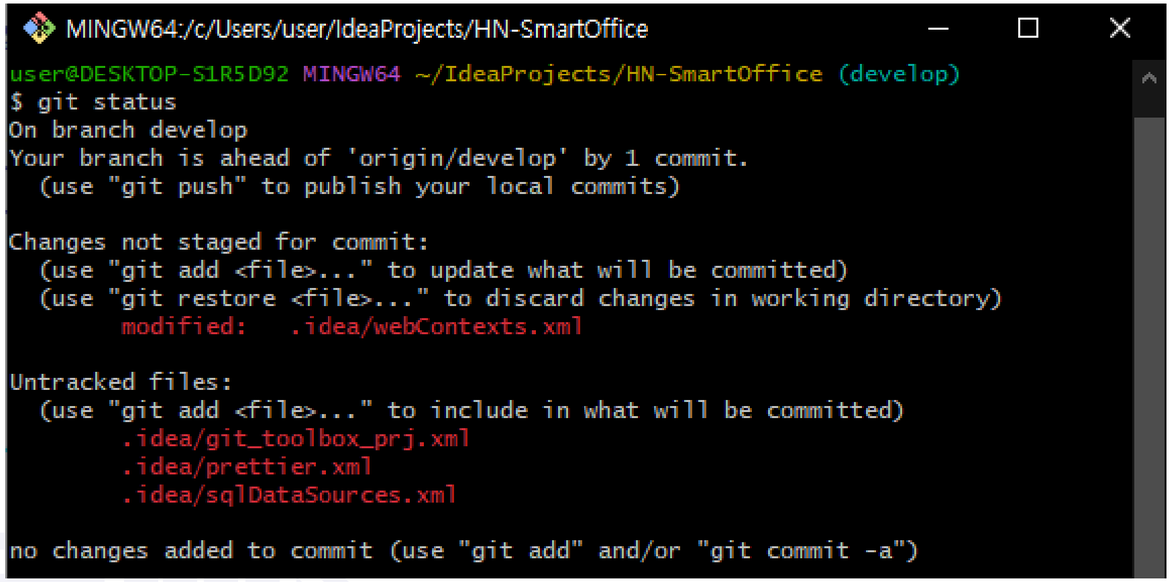
용어 소개

용어

Repository	프로젝트의 모든 파일과 폴더의 변경 이력을 저장하는 곳 GIT은 원격저장소(Remote Repository)와 로컬 저장소(Remote Repository) 두 종류의 저장소를 제공한다
Remote Repository	파일이 원격 저장소 전용 서버에서 관리되며 여러 사람이 함께 공유하기 위한 저장소다.
Local Repository	원격 저장소. 파일을 원격 저장소 전용 서버에서 관리하고 공유가 가능하다.
SnapShot	특정 시점에서 파일, 폴더 또는 워크스페이스의 상태 이러한 스냅샷은 커밋(Commit)이라고도 불린다
commit	변경 사항을 저장하는 작업입니다. 커밋은 프로젝트의 특정 시점에서의 상태를 스냅샷으로 기록하며, 변경 이력을 추적하는 데 사용됩니다.

용어

Working Directory	현재 작업 중인 프로젝트의 파일들이 있는 디렉터리 이 디렉터리는 Git 저장소의 일부이며, 프로젝트의 파일을 수정하고 변경하는 작업을 수행합니다. Working Tree(작업 트리) , Working Copy(작업 복사본)라고도 합니다.



파일관리(추적)의 상태

추적안함 (Untracked) : 관리대상이 아님

추적함(Tracked)

- 수정없음 (Unmodified) : 변경이 없는 파일
- 수정함 (Modified) : 변경된 파일
- 스테이지됨 (Staged) : 스테이지에 올라간 파일

git staus 를 사용하여 현재 작업 디렉토리의 상황을 확인 가능합니다.

용어

Staging	커밋할 변경 사항을 준비하는 단계를 의미합니다. 변경된 파일 중 일부만을 선택하여 스테이징할 수도 있습니다. 작업 순서 1. Working Tree에서 변경된 파일을 선택하여 스테이징 영역(Staging Area)으로 추가. 2. 스테이징된 파일들은 다음 커밋에 포함될 준비가 되어 있음을 나타냅니다. 3. 커밋을 수행하여 변경 사항을 영구적으로 저장.

작업 순서

1. 변경 사항 확인:
- 현재 작업 중인 디렉토리에서 git status 명령을 사용하여 변경된 파일 목록을 확인합니다.
- 이 명령은 작업 트리와 스테이징 영역 간의 차이점을 보여줍니다.

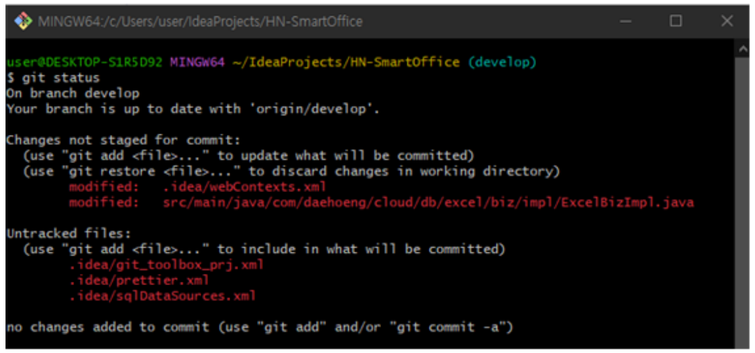
2.파일 스테이징:
- 변경된 파일 중 스테이징하고자 하는 파일을 선택합니다.
- 선택한 파일을 스테이징하려면 git add <파일명> 명령을 사용합니다.
- 혹은 모든 변경된 파일을 스테이징하려면 git add . 명령을 사용할 수 있습니다.

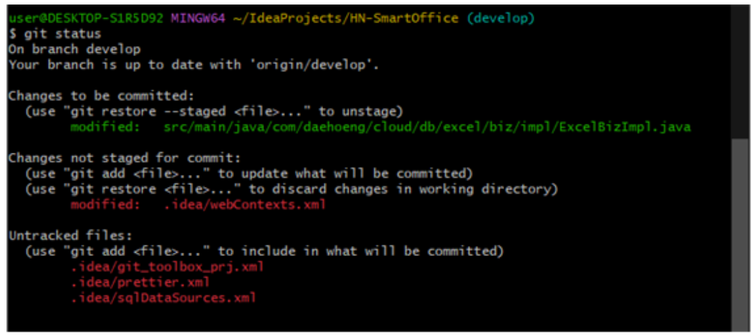
3.스테이징 확인:
- 파일을 스테이징한 후에는 git status 명령을 다시 실행하여 스테이징된 파일 목록을 확인합니다.
- 스테이징된 파일은 "Changes to be committed" 또는 "Changes staged for commit" 섹션에 나타납니다.

4. 커밋:
- 변경 사항을 스테이징한 후, 이제 커밋을 수행하여 변경 사항을 영구적으로 저장할 준비를 합니다.
- git commit 명령을 사용하여 커밋을 수행합니다.
- 커밋 메시지를 작성하고 저장하면 변경 사항이 로컬 저장소에 영구적으로 기록됩니다.

용어

Staging + commit 예시





작업 순서

1. 변경 사항 확인:

Changes not staged for commit:

git add를 사용하여 스테이징되지 않은 변경 사항입니다.

.idea/webContexts.xml 및 src/main/java/com/daehoeng/cloud/db/excel/biz/impl/ExcelBizImpl.java 파일이 수정되었습니다.

Untracked files:

아직 Git에서 추적되지 않은 파일입니다.

.idea/git_toolbox_prj.xml, .idea/prettier.xml, .idea/sqlDataSources.xml 파일이 새로 추가되었습니다.2.파일 스테이징:

- 변경된 파일 중 스테이징하고자 하는 파일을 선택합니다.
- 선택한 파일을 스테이징하려면 git add <파일명> 명령을 사용합니다.
- 혹은 모든 변경된 파일을 스테이징하려면 git add . 명령을 사용할 수 있습니다.

2.파일 스테이징:

git add 명령을 사용하여 ExcelBizImpl.java 파일을 스테이징했습니다.

이제 이 파일의 변경 사항이 다음 커밋에 포함될 준비가 되었습니다.

3.스테이징 확인:

- 파일을 스테이징한 후에는 git status 명령을 다시 실행하여 스테이징된 파일 목록을 확인합니다.

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

modified: src/main/java/com/daehoeng/cloud/db/excel/biz/impl/ExcelBizImpl.java

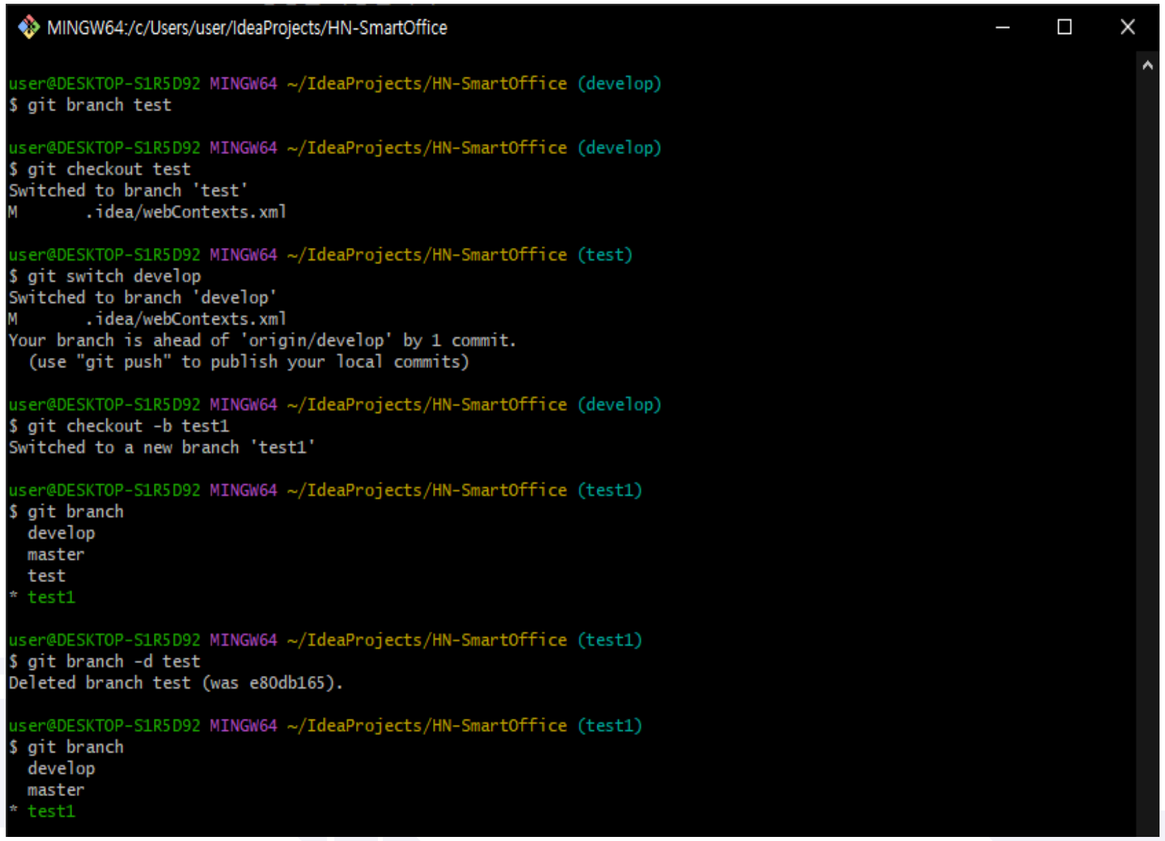
4. 커밋:

- 변경 사항을 스테이징한 후, 이제 커밋을 수행하여 변경 사항을 영구적으로 저장할 준비를 합니다.
- git commit 명령을 사용하여 커밋을 수행합니다.
- 커밋 메시지를 작성하고 저장하면 변경 사항이 로컬 저장소에 영구적으로 기록됩니다.
- git commit -m "test"

용어

barnch

Git에서 개발 작업을 분리하고 관리하기 위한 독립적인 작업 공간입니다.
가지 또는 분기점을 의미하며, 작업을 할때에 현재 상태를 복사하여 Branch에서 작업을 한 후에 완전하다 싶을때 Merge 하여 작업을 진행한다.
브랜치는 개발 작업을 분리하여 동시에 여러 기능을 개발하고 서로 충돌하지 않도록 합니다. 유연하고 효과적인 개발 및 협업을 가능하게 합니다.



1. 브랜치 생성:
새로운 기능을 개발하거나 버그를 수정하는 등의 작업을 할 때 새로운 브랜치를 만듭니다.
새로운 브랜치를 만들려면 git branch <branch_name> 명령을 사용합니다.

2. 브랜치 전환:
현재 작업 중인 브랜치를 변경하려면 git checkout <branch_name> 명령을 사용합니다.
Git 2.23 버전 이후부터는 git switch 명령을 사용하여 브랜치를 전환할 수도 있습니다.

3. 브랜치 생성 및 전환:
새로운 브랜치를 만들고 동시에 그 브랜치로 전환하려면 git checkout -b <branch_name> 명령을 사용

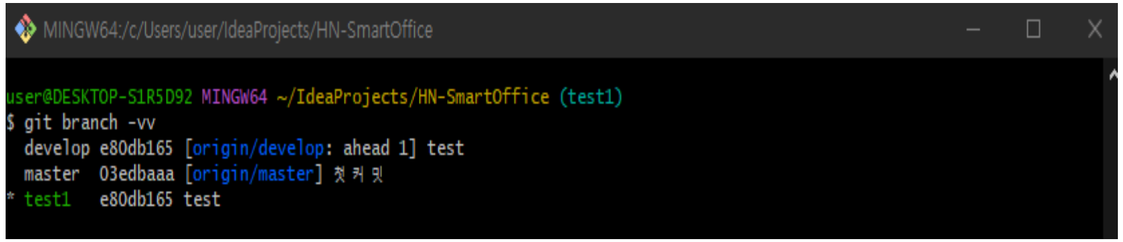
4. 브랜치 목록 확인:
현재 저장소에 있는 모든 브랜치 목록을 확인하려면 git branch 명령을 사용합니다.
현재 브랜치에 표시된 별표(*)로 현재 작업 중인 브랜치를 확인할 수 있습니다.

5. 브랜치 삭제
로컬에서 브랜치를 삭제하려면 git branch -d <branch_name> 명령을 사용합니다.

용어

Upstream barnch

로컬 브랜치가 연결된 원격 저장소의 브랜치를 가리킵니다.
업스트림 브랜치를 설정하면 Git은 해당 브랜치의 변경 사항을 원격 저장소로 푸시하거나 원격 저장소의 변경 사항을 가져와 로컬 브랜치로 풀(fetch) 수 있습니다.



업스트림 브랜치 확인

git branch -vv

1. develop 브랜치:
현재 test 브랜치로부터 하나의 커밋을 앞서고 있습니다.
원격 저장소의 origin/develop 브랜치를 업스트림으로 설정하고 있습니다.
2. master 브랜치:
로컬의 master 브랜치는 원격 저장소의 origin/master 브랜치를 업스트림으로 설정하고 있습니다.
3. test1 브랜치:
현재 test 브랜치로부터 하나의 커밋을 앞서고 있습니다.
로컬의 test1 브랜치는 원격 저장소와의 연관이 없는 것으로 보입니다.

업스트림 브랜치 푸시

git push -u <원격저장소명> <로컬브랜치명>

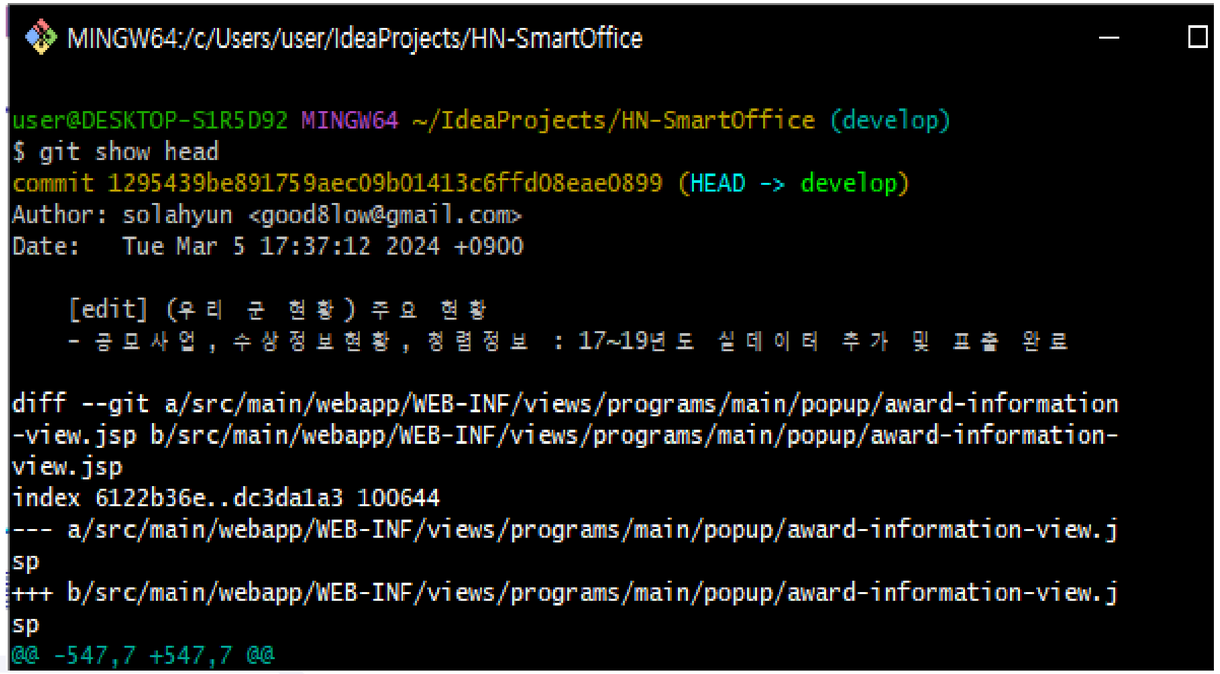
해당 로컬 브랜치와 원격 저장소의 브랜치 간에 동기화를 수행할 때 사용

용어

master barnch	기본적으로 Git 저장소가 생성될 때 생성되는 기본 브랜치입니다. 주로 안정적인 상태의 코드를 유지하는 데 사용됩니다. 즉, 배포 가능한 코드가 있는 브랜치입니다. 보통 프로덕션 환경에서 실행되는 코드가 master 브랜치에 있습니다.
feature barnch	새로운 기능을 개발하기 위해 만들어지는 브랜치입니다. 주로 기능 개발 또는 버그 수정을 위한 작업을 진행합니다. feature 브랜치에서 개발이 완료되면, master 브랜치로 병합하여 새로운 기능이나 수정이 반영됩니다. 각각의 기능 또는 작업마다 별도의 feature 브랜치를 생성하여 개발을 진행합니다.

용어

Head	Git에서 현재 작업 중인 위치를 가리키는 포인터입니다. 역할 1. 현재 작업 중인 브랜치를 가리킵니다. 2. 다음 커밋이 이루어질 위치를 가리킵니다. 3. 작업 트리에 보여지는 커밋의 상태를 결정합니다.



Git head를 확인하는 명령어인 git show head 명령시 결과물 입니다.

설명

커밋 해시: 1295439be891759aec09b01413c6ffd08eae0899

브랜치: develop

작성자: solahyun good8low@gmail.com

작성일: Tue Mar 5 17:37:12 2024 +0900

커밋 메시지: "[edit] (우리 군 현황) 주요 현황 - 공모사업, 수상정보현황, 청렴정보 : 17~19년도 실데이터 추가 및 표출 완료"

용어

Commit Hash

Git에서 커밋을 고유하게 식별하는 문자열입니다.
해시는 Git 저장소에서 특정 커밋을 식별하는 데 사용됩니다.
주로 SHA-1 해시 알고리즘을 사용하여 커밋 해시를 생성합니다.
일반적으로 커밋 해시는 40자의 16진수 문자열로 표현됩니다.

커밋 해시 확인
git log 명령을 사용

커밋 해시: 7ccfbda1837825855160e027b4fdc5bb9e3051d7
브랜치: origin/develop
작성자: solahyun good8low@gmail.com
작성일: Mon Mar 11 17:15:58 2024 +0900
커밋 메시지:
"[edit] 고향사랑기부현황(금액별) -
금일을 어제 날짜로 수정, 날짜 기준 어제로 수정"

용어

Merge	두 개의 다른 브랜치를 하나로 통합하는 Git 작업입니다.

1. 기능 브랜치를 메인 브랜치로 통합:
새로운 기능을 개발하기 위해 만든 브랜치에서 작업한 후, 이를 메인 브랜치(예: master 또는 develop)로 병합합니다.
이를 통해 새로운 기능이나 변경 사항을 메인 브랜치로 통합하여 프로젝트의 다른 부분과 통합할 수 있습니다.

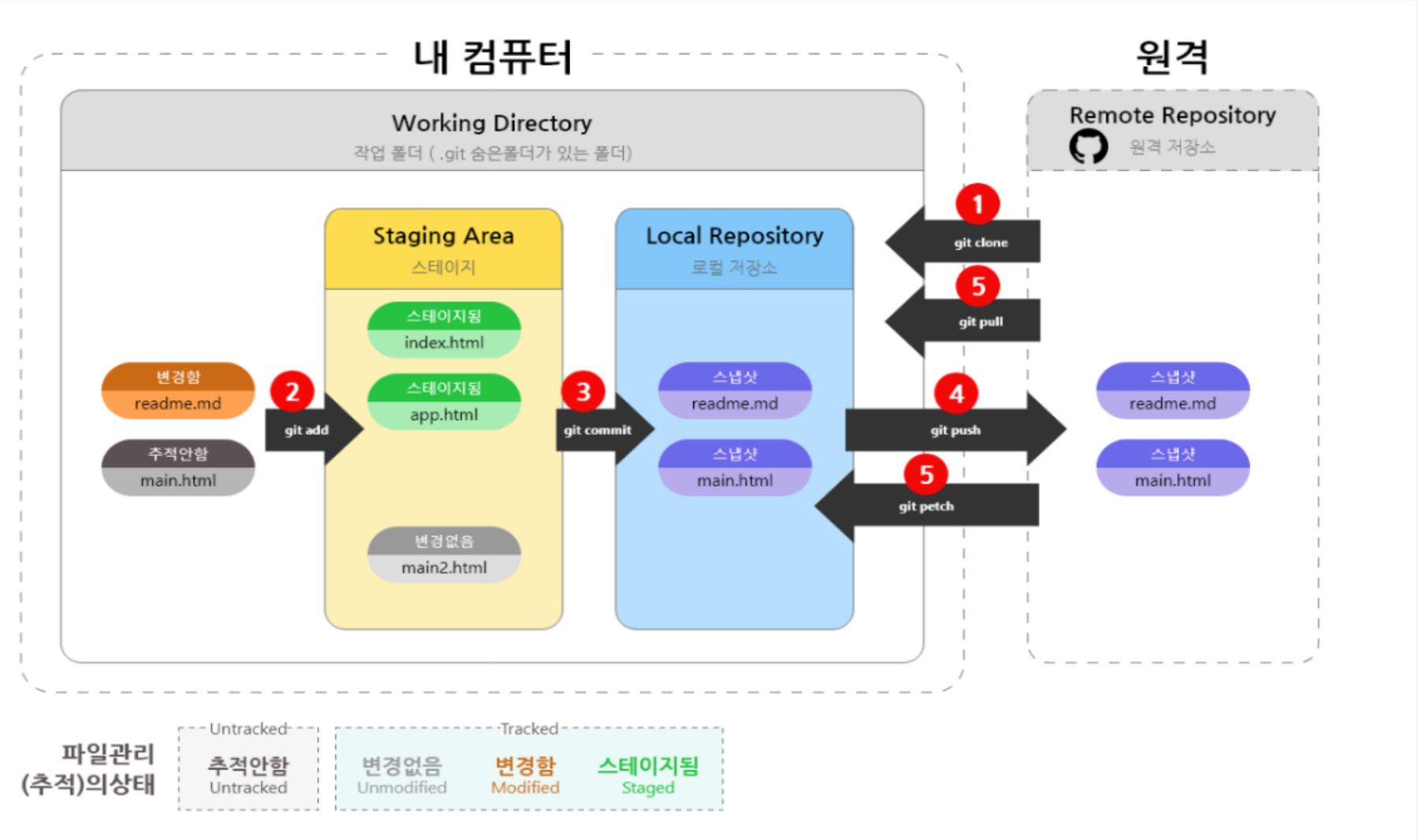
2. 두 개의 다른 브랜치를 병합:
서로 다른 개발자들이 동시에 작업한 두 브랜치를 병합하여 작업한 내용을 통합합니다.
이를 통해 서로 다른 작업을 한 개발자들이 작업한 내용을 통합할 수 있습니다.

git merge <other_branch>를 통해서 병합

Chapter 3.

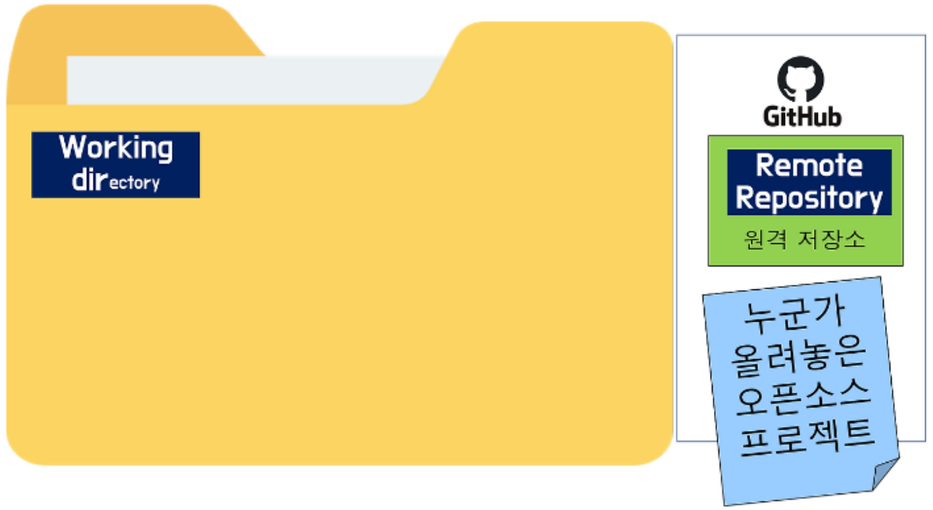
구조 소개

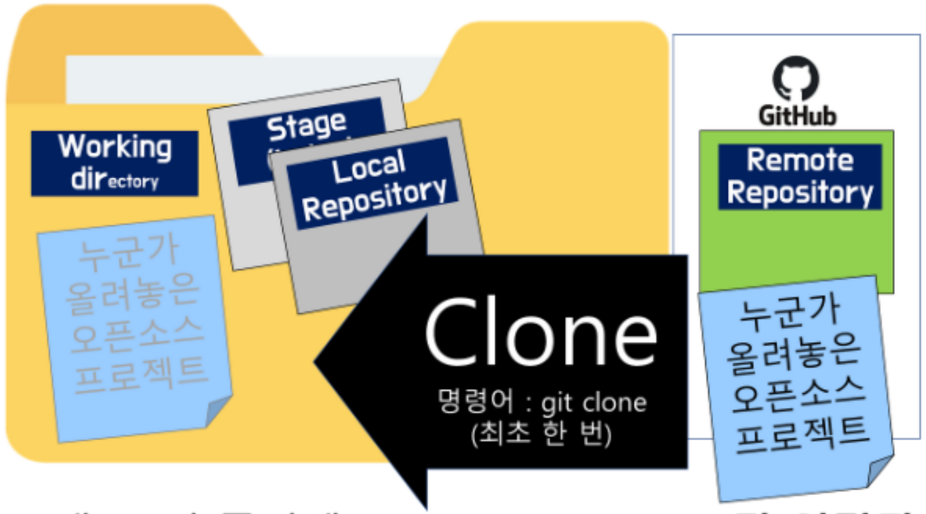
구조



출처 : https://inpa.tistory.com/entry/GIT-%E2%9A%A1%EF%B8%8F-%EA%B0%9C%EB%85%90-%EC%9B%90%EB%A6%AC-%EC%89%BD%EA%B2%8C%EC%9D%B4%ED%95%B4

GIT CLONE 프로세스

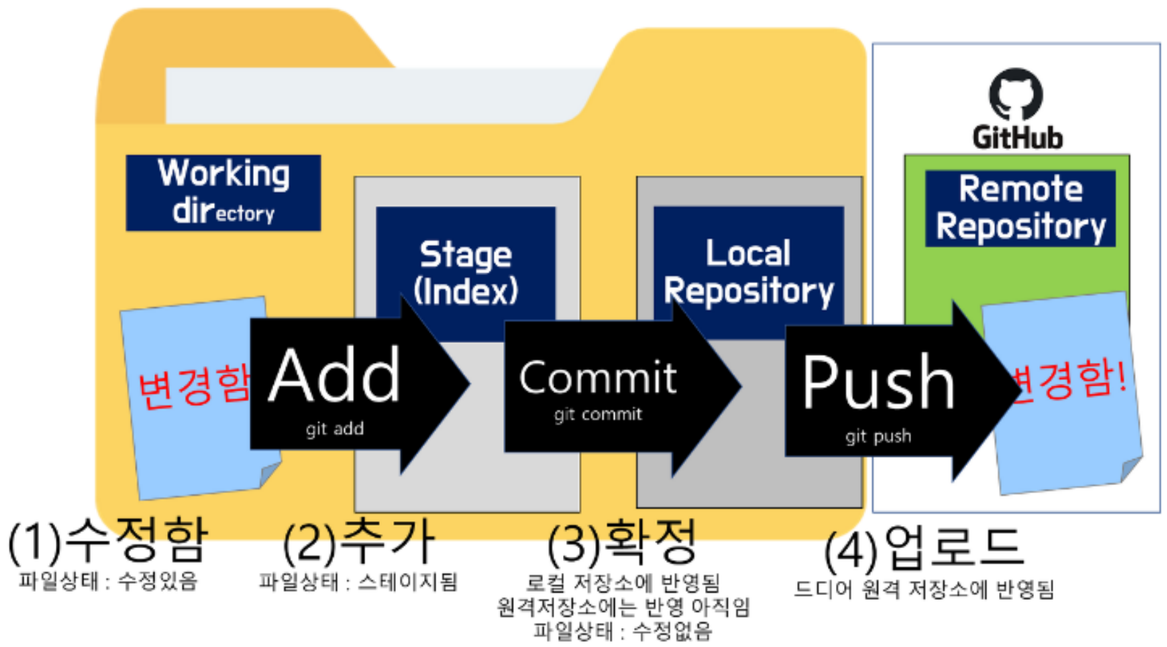




: 내 PC의 폴더에 **Remote repository랑 연결된**
(원격저장소)
Local repository가 생성됨!
(로컬저장소)

출처 : https://pangguinland.tistory.com/206

GIT PUSH 프로세스



출처 : https://pangguinland.tistory.com/206

1. **수정** — 파일을 수정합니다.
2. **추가** — 수정된 파일을 Staging 합니다
3. **확정** — Stage에 올라간 파일을 로컬 브랜치에 커밋
4. **업로드** — 로컬브랜치에 커밋된 파일을 원격브랜치에 푸시

Chapter 3.

명령어 소개

명령어

git clone

특정 git repository를 가져옵니다.

git init

프로젝트(소스코드들이 있는 디렉토리)를 git repository로 만들기 위해서 사용하는 명령어 입니다.

디렉토리를 git repository로 만들어야 git으로 버전 관리를 할 수 있습니다.

git branch

branch를 생성할 때 사용됩니다.

git checkout

branch를 checkout 할때 사용되는 명령어입니다.

git add

파일의 수정 사항들(Modified)을 staged 상태로 변경할 때 사용하는 명령어 입니다.

git repository에 새로 추가된 파일들을 staged 상태로 옮길때도 사용됩니다.

새로이 추가된 파일들은 "untracked" 파일 이라고 하는데, git에서는 이들도 수정 사항이라고 인식합니다.

git commit

staged 된 파일들을 commit 하고자 할때 사용하는 명령어 입니다.

git push

branch에서 commit한 파일들을 원격 저장소로 보내 동일한 상태로 만들어줍니다.

명령어

git diff

어떤 수정 사항들이 적용됐는지 볼 때 사용하는 명령어 입니다.

참고로 staged 된 수정 사항들은 git diff로 볼 수 없습니다.

Modified 된 파일들만 git diff로 볼 수 있습니다.

git status

현재 상태를 보여주는 명령어 입니다.

어떠한 파일들이 modified가 되었고 어떠한 파일들이 staged가 되었는지 등의 전체적인 상황을 보여줍니다.

git log

Commit 내역들을 보여줍니다.

Commit history라고도 합니다. git log를 통해 이제까지 커밋 내역들을 전부 볼 수 있습니다.

다만 출력되는 포맷이 보기가 쉽지가 않아서 tig 같은 tool을 사용하면 훨씬 편리합니다.

git rm

원하는 파일을 git repository에서 삭제합니다.

git mv

원하는 파일을 git repository 상에서 이동 시킬때 사용합니다.

파일의 이름을 바꿀 수도 있습니다.

Chapter 5.

웹서비스

웹서비스

GitHub

소프트웨어 개발 프로젝트의 버전 관리와 협업을 위한 웹 기반 플랫폼입니다.
Git 버전 관리 시스템을 기반으로 하며, 개발자들이 코드 변경 사항을 추적하고, 다른 사람들과 협업하며,
소프트웨어의 다른 버전을 효율적으로 관리할 수 있도록 합니다.

GitLab

Git 버전 관리 시스템을 기반으로 하며, 소프트웨어 개발 및 프로젝트 관리에 유용한 다양한 도구를 제공합니다.
GitLab은 개방 소스 코드로 구축되어 있어 기업 내에서 자체 호스팅할 수도 있습니다.

GitTea

오픈 소스로 개발된 자체 호스팅형 Git 서비스 플랫폼입니다.
GitHub나 GitLab과 같이 코드 저장소를 관리하고 협업할 수 있는 기능을 제공하지만, 상대적으로 더 경량화되고 간단한 사용자 경험을 제공하는 것이 특징입니다.

웹서비스

서비스 별 구분

기능/특징	GitHub	GitLab	GitTea
자체 호스팅 옵션	제한적	풍부	가능
커뮤니티 크기	대규모	중간	작음
무료 플랜	제공 (일부 제한)	풍부	제공
유료 요금제	있음	있음	없음
CI/CD 기능	제공 (GitHub Actions)	제공	제한적
보안 및 규정 준수	일부 제공	강화	-
기능 다양성	풍부	풍부	제한적
사용자 인터페이스	풍부	풍부	간소화
오픈 소스	아니요	예	예
커스터마이징 가능성	제한적	높음	높음

이상 발표를 마무리 하겠습니다

감사합니다